Measuring Cooperative Robotic Systems Using Simulation-Based Virtual Environment

Xiaolin Hu
Computer Science Department
Georgia State University, Atlanta GA, USA 30303

Bernard P. Zeigler
Arizona Center for Integrative Modeling and Simulation
University of Arizona, Tucson AZ, USA 85721

ABSTRACT

Simulation-based study plays an important role in experimenting, understanding, and evaluating intelligent robotic systems. While robot models can be created and studied in a simulated environment, replacing some of the robot models with their real robot counterparts brings simulation-based study one step closer to the reality. It also provides the flexibility to allow real robots to be experimented within a virtual environment. This capability of robot-in-the-loop simulation is especially useful for large-scale cooperative robotic systems whose complexity and scalability severely limit the possibility for study and evaluation in a physical environment with real robots. This paper presents a simulation-based approach that allows a cooperative robotic system to be effectively evaluated in a virtual environment with combined real and virtual robots. This capability adds to conventional simulation-based study to form an integrated measuring process. An example of robotic convoy system is presented together with metrics to measure the formation coherence of cooperative robotic system. Some preliminary simulation results are presented.

KEYWORDS: *Cooperative Robotic System, Virtual Environment, Robot-in-the-Loop Simulation, Robotic Convoy System*

1. INTRODUCTION

Cooperative robotic systems couple computational intelligence to the physical world. These systems consist of multiple homogenous or heterogeneous robots that perceive the environment, make decisions, and carry out commands to affect the environment. Communication and cooperation is important for theses systems since their robots work as a collective team to finish common tasks. Several taxonomies and metrics have been defined to classify these systems. For example, Dudek, *etc.* [1] classifies robotic collectives along seven dimensions: size of the collective, communication range, communication topology, communication bandwidth, collective reconfigurability, processing ability of each collective unit, and collective composition. Balch [2] classifies the performance metric of multirobot tasks based on time, subject of action, resource limits, group movement, platform capabilities, *etc*.

The increasing complexity of collective robotic systems calls for systematic methods as well as supporting environments to experiment, understand, and evaluate these systems. To serve this purpose, modeling and simulation technologies are frequently applied. With simulation-based methods, models of robots can be built and simulated. Different configurations can be easily applied to experiment and measure the performance of the system under development. To allow simulation of robotic systems that actively interact with an external environment, an environment model needs to be created. This environment model serves as a "virtual" environment to provide sensory input to robot models and to response to robots' actuation. For example, a virtual environment for mobile robots simulation can have virtual obstacles that can be sensed by robot models, and it responds to robots' movements by updating new sensory information to robot models.

While robot models can be created and studied in a simulated environment, replacing some of the robot models with their real robot counterparts will bring simulation-based study one step closer to the reality and provides the flexibility to allows real robots to be experimented in a virtual environment. This capability of robot-in-the-loop simulation is especially useful for large-scale cooperative robotic systems whose complexity and scalability severely limit experimentation in a physical environment using all real robots. This paper presents an approach that allows a cooperative robotic system to be effectively evaluated in a virtual environment with combined real and virtual robots. This research is an extension to our previous work on a simulation-based software development methodology for cooperative robotic systems [3, 4]. This methodology supports "model continuity" so the simulation models in the design stage can be directly mapped to real robots to control the real robots in execution. It greatly eases the transition from simulation-based study to real robot implementation and

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **AUG 2004**	2. REPORT TYPE	3. DATES COVERED **00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE **Measuring Cooperative Robotic Systems Using Simulation-Based Virtual Environment**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Georgia State University,Computer Science Department,Atlanta,GA,30303**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the 2004 Performance Metrics for Intelligent Systems Workshop (PerMIS -04), Gaithersburg, MD on August 24-26 2004

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**7**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

increases the confidence that the final system implements the behavior as been developed. This research is based on the Discrete Event System Specification (DEVS) modeling and simulation framework [5].

The concept of virtual environment has been largely used by the technology of virtual reality (VR), which has been applied to various areas such as simulation of manufacturing plants, the planning of robotic workcells, and robot teleoperation systems. While the research of VR mainly deals with the interaction with human operators, our work focuses on the interaction between robots and the virtual environment. The following research work is related to our research from this perspective. Komoriya and Tani [6] developed a virtual environment that allows a single real robot to be experimented in a virtual environment. Wang [7] proposed a simulation environment that allow real and virtual robot to work together. The work of RAVE [8] developed a simulation environment that supports multiple mobile robotic systems. Our research extends these works by developing a well-defined architecture, an incremental development process, and by integrating experimental frames to measure cooperative robotic systems with combined real and virtual robots in a systematic way.

This paper is organized as follows. Section 2 presents the virtual measuring environment from three aspects: the architecture, the measuring process, and the relationship to experimental frames. Section 3 describes a robotic convoy system as an illustrative example. The models of this system are first described, several metrics are then presented, and some preliminary simulation data is given. Section 4 concludes this work and provides future research directions.

2. A VIRTUAL EVALUATION ENVIRONMENT FOR ROBOTIC SYSTEMS

The effectiveness of this simulation-based virtual evaluation environment is supported by a well-defined architecture, an incremental measuring process, and by integrating experimental frames to specify metrics for performance measurement. Next we present these three aspects respectively.

2.1 Architecture of the Virtual Evaluation Environment

In this research we view robotic systems as a particular form of real-time systems that monitor, respond to, or control, an external environment. This environment is connected to the computer system through sensors, actuators, and other input-output interfaces [9]. A robotic system from this point of view consists of sensors, actuators and the decision-making unit. A cooperative robotic system is composed of a collection of robots that communicate with each other and interact with an environment.

This above description suggests the basic structure for a simulation-based virtual environment for robotic systems: an environment model, and a collection of robot models that include a decision making model, sensors, and actuators. The environment model represents the real environment within which the robotic system will be executed. It may include virtual obstacles, virtual robots, or any other entities that are useful for simulation-based study. It forms a virtual environment for the robots. The robot model represents the control software that governs the robot's behavior. It also includes sensor and actuator interfaces to bridge the decision-making model and the simulation-based virtual environment. In our research, we clearly separate a robot's decision-making unit, which is modeled as a DEVS atomic or coupled model, from the sensors and actuators that are modeled as DEVS Activities. Couplings can be added between DEVS Activities and the environment model thus messages can be passed between the decision-making model and the environment model through sensor/actuator Activities.

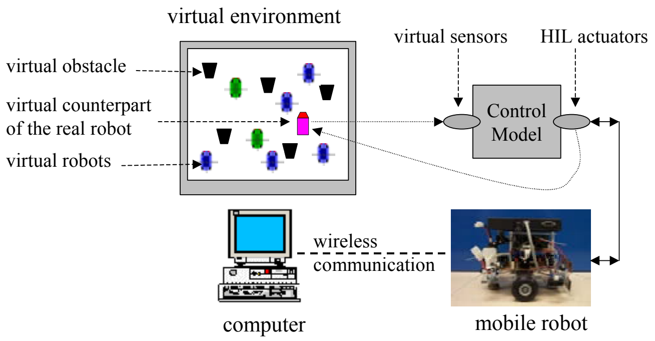


Figure 1: Architecture of robot-in-the-loop simulation

The clear separation between robots' decision-making model and sensor/actuator interfaces brings several advantages. First, it separates a robot's decision-making from hardware interaction, thus making it easier for the designer to focus on the decision-making model, which is the main design interest. Secondly, the existence of a sensor/actuator interface layer makes it possible for the decision-making model to interact with different types of sensors/actuators, as long as the interface functions between them are maintained the same. Thus depending on different experimental and measuring objectives, a robot model can be equipped with different sensors/actuators to be experimented and measured. Our previous work [10] has taken advantage of these features to allow direct transferring of the decision-making models from simulation to real robots execution – a capability referred as model continuity. During simulation, the decision-making model interacts with a virtual environment through virtual sensors/actuators; during real execution, the real robots' decision-making models interacts with a real environment through real sensor/actuator interfaces. An intermediate stage can also be developed to allow the decision-making model on a real robot to interact with the simulation-based virtual environment. We call this stage robot-in-the-loop simulation. It is achieved by configuring a real robot to use a combination of virtual and real sensors/actuators. For example, Figure 1 shows an experimental setup where one real mobile robot works together with a virtual environment. In this example, the mobile robot uses its virtual sensor to get sensory input from the virtual environment and uses its real motor interface to move the robot. As a result, this real robot moves in a physical field based the sensory input from a virtual environment. Within this virtual environment, the robot can "see" virtual obstacles and other virtual robots that are simulated by computers. This capability of robot-in-the-loop simulation brings simulation-based study one step closer to the reality. It also makes it possible to study and measure several real robots within a large robotic system that may include hundreds of robots. In this case, the rest of robots can be provided by the simulation-based virtual environment.

One important issue for the robot-in-the-loop simulation is the synchronization between the real robots and the virtual environment. For example, in Figure 1, when the decision-making model issues a moving command, the real robot will move a distance in the physical environment. This change of position should also be updated by the virtual environment. For this purpose, each real robot has a virtual counterpart in the virtual environment. When a real robot moves, the position of its virtual counterpart will be updated. Thus the synchronization between the real robot and the virtual environment is actually the synchronization between the real robot and its virtual counterpart. Ideally, an independent monitoring system is needed to track the movement of the real robots and then inform the virtual environment to synchronize the distance and time of robots' movements. In our current implementation, a set of HIL (hardware-in-the-loop) sensors/actuators has been developed. These HIL sensors/actuators drive the real sensor/actuators, while in the meantime are coupled to the virtual environment thus messages can be sent to it. For the example shown in Figure 1, the HIL motor drives the motor of the robots. In the meantime it catches the *moveComplete* signal returned from the motor and then sends a message to the virtual environment to update the position of its virtual counterpart.

2.2 From Robot Model To Real Robot – An Incremental Measuring Process

Based on this virtual measuring environment, an incremental measuring process is developed. This process includes three steps and supports smooth transitions between them. These steps are measuring based on conventional simulation, measuring based on robot-in-the loop simulation, and measuring based on real robot execution. Figure 2 gives an example with two robots to illustrate this process.

The first step is conventional simulation, where all components are models and simulated by fast-mode or real-time simulators in one computer. As shown in Figure 2(a), both robot models are equipped with virtual sensors and actuators to interact with the virtual environment. Couplings between two robots can also be added so they can send messages to each other. We note that this is the same setup as the simulations that most robotic research uses. It has the most flexibility as all components are models and different configurations can be easily applied to measure the system under development.

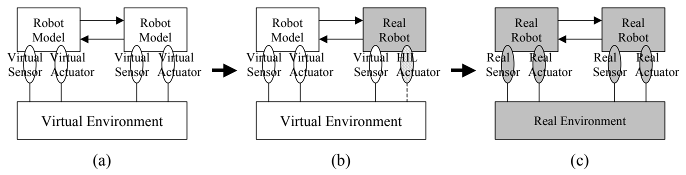


Figure 2: An incremental measuring process

The second step is robot-in-the-loop simulation where one or more real robots are measured and experimented within a virtual environment together with other virtual robots (robot models) that are simulated by computer. In this step, the virtual robots still use virtual sensors/actuators. However, depending on the measuring objectives, the real robots may have a combination of virtual and HIL sensors/actuators. For example, the real robot shown in Figure 2(b) uses a virtual sensor and a HIL actuator. The couplings between the two robots are maintained the same so the real and virtual robots can interact with each other in the same way as in the first step. However, the real commutation in this step happens across a wireless network, which is transparent to the robots. Since real robots are involved in the simulation, robot-in-the-loop simulation has to run in a real-time fashion.

The final step is the real system measurement, where real robots are measured in a real physical environment. These robots use real sensors and actuators. They communicate in the same way as the first two steps since the couplings between them are not changed through the process. The measurement of this step is from the reality, thus having the most fidelity. However, it is also most costly and time consuming among the three steps.

This incremental measuring process brings simulation-based study closer and closer to the reality. As the process proceeds, the flexibility (easy to experiment different configurations) and productivity (time saving and cost saving) of the measurement decreases and the fidelity (loyal to the reality) of the measurement increases.

2.3. Specify Measuring Metrics Using Experimental Frame

An experimental frame is a specification of the conditions within which the system is observed or experimented [5]. In DEVS-based modeling and simulation framework, an experimental frame is realized as a system that interacts with the source system, or System Under Test (SUT), to obtain the data of interest under specified conditions. It consists of four major subsections:

- *input stimuli*: specification of the class of admissible input time-dependent stimuli. This is the class from which individual samples will be drawn and injected into the model or system under test for particular experiments.

- *control*: specification of the conditions under which the model or system will be initialized, continued under examination, and terminated.
- *metrics*: specification of the data summarization functions and the measures to be employed to provide quantitative or qualitative measures of the input/output behavior of the model. Examples of such metrics are performance indices, goodness-of-fit criteria, and error accuracy bound.
- *analysis*: specification of means by which the results of data collection in the frame will be analyzed to arrive at final conclusions. The data collected in a frame consists of pairs of input/output time functions.

When an experimental frame is realized as a system to interact with the SUT (or its model), the four specifications become components of the driving system. For example, a generator of output time functions implements the class of input stimuli.

Integrate experimental frames into the virtual measuring environment brings the advantage that measuring metrics can be formally specified. More research is on the way to integrate them in a structured way. In the meantime a set of measuring metrics is also under development for cooperative robotic systems.

3. ROBOT CONVOY: A CASE STUDY EXAMPLE

The presented virtual measuring environment has supported the development of a robotic convoy system. Below we briefly describe the model of this system, its measuring metrics, and some preliminary results that are collected from simulation-based study. We note that most results presented in this paper are collected from simulations that do not involve real robots. But in the next step we plan to measure the system using robot-in-the-loop simulation and expect to reach more interesting results. For example, we plan to use one real robot to run robot-in-the-loop simulation to check the convoy speed of this real robot and compare it with the data collected from the conventional simulation. Another result that we plan to check is to use two real robots neighboring to each other and then check the back robot's position errors based on the position and direction of its front robot in the physical environment.

3.1 System Description and System Model

This robot convoy system consists of an indefinite number of robots, saying N robots (N>1). These robots are in a line formation where each robot (except the leader and the ender) has a front neighbor and a back neighbor. The robots used in this system are car type mobile robots with wireless communication capability. They can move forward/backward and rotate around the center, and have whisker sensors and infrared sensors. [11].

One of the basic goals of this convoy system is to maintain the coherence of the line formation and to synchronize robots' movements. Synchronization means a robot cannot move forward if its "front" robot doesn't move, and it has to wait if its "back" robot doesn't catch up. To serve this purpose, synchronization messages are passed between a robot and its neighbors. To achieve coherence of the line formation, the moving parameters of a "front" robot are passed back. This allows the back robot to plan its own movement accordingly based on its front robot's movement. The system has no global communication and coordination since we want to study how global behavior can be achieved using localized sensing and communication.

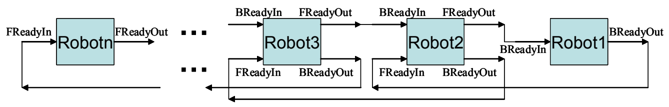


Figure 3: System model of the robotic convoy system

Figure 3 shows the model of this system. As we can see, this model includes N models (each of them is a DEVS coupled model), which are corresponding to the N robots in the system. Each intermediate robot model has two input ports: *FReadyIn*, *BReadyIn* and two output ports: *FReadyOut*, *BReadyOut*. These ports are used to send and/or receive synchronization messages between robots and to pass moving parameters from a "front" robot to the "back" robot. The couplings between them are shown in Figure 3.

During the convoy, the *leader* robot (*Robot1* in Figure 3) decides the path of convoy. Meanwhile, it will turn around if its infrared sensors indicate that there are obstacles ahead. All other robots conduct movement based on their sensory input and the moving parameters passed back from their front robots. Specifically, a robot will "predict" where its front robot is and turn to that direction. It then moves forward or backward to "catch" its front robot. After that it may go through an "adjust" process to make sure that it does not lose its front robot. This adjust process is necessary because noise and variance exist during a movement so a robot will not reach the desired position and direction after the movement. During adjustment, a robot "scans" around until it finds its front robot. Then it sends out a synchronization message to "inform" its front and back neighbors. Thus robots actually go through a basic "turn—move—adjust—inform" routine. For example, a robot R_{i-1} will turn angle α_{i-1} to the direction of its front robot R_{i-2}, move distance d_{i-1} to "catch" its front robot, and then adjust itself with angle β_{i-1} to make sure it "sees" its front robot R_{i-2}. Figure 4 shows these moving parameters.

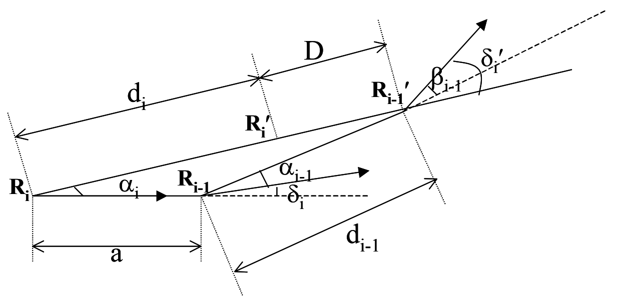


Figure 4: Moving parameters for robots' convoy

After the adjustment, R_{i-1} sends out a synchronization message to its neighbors. This synchronization message contains information of α_{i-1}, d_{i-1}, and β_{i-1}. Based on this information and its sensory data, R_i plans its movement. This is shown by Figure 4 and formulated by formula (1)-(3). Among these formulas, δ_i is the angle (direction) difference between R_i and R_{i-1}; a is the distance between robot R_i and R_{i-1} and can be calculated from the robot's infrared sensor data and the size of the robot. Specifically, the turning angle α_i of R_i is calculated by formula (1); the moving distance d_i can be calculated from formula (2), where D is the desired distance between R_i and R_{i-1}. Then the new angle difference δ_i' between R_i and R_{i-1} is updated by formula (3), where β_i is the adjusting angle for R_i. We note that due to noise and variance, the δ_i' calculated from formula (3) will not be the exact angle difference between R_i and R_{i-1}. However, it seems that this error does not accumulate as time proceeds.

$$tg\alpha_i = \frac{d_{i-1} * \sin(\alpha_{i-1} + \delta_i)}{a + d_{i-1} * \cos(\alpha_{i-1} + \delta_i)} \qquad (1)$$

$$d_i = \frac{d_{i-1} * \sin(\alpha_{i-1} + \delta_i)}{\sin \alpha_i} - D \qquad (2)$$

$$\delta_i' = \delta_i + \alpha_{i-1} + \beta_{i-1} - \alpha_i - \beta_i \qquad (3)$$

The model of each robot is developed based on the subsumption architecture [12]. It has the *Avoid* model to avoid collisions with any objects; the *Convoy* model to control robot's movement based on the rules as described above. It also has DEVE Activities to represent the sensor/actuator interfaces of the robot. A detailed description of a similar model can be found at [3].

Figure 5 shows the *Environment* model that we used for this example. This *Environment* model includes *TimeManager* models and the *SpaceManager* model. For each robot, there is a *TimeManager* corresponding to it. This *TimeManager* models the time for a robot to conduct a movement. The *SpaceManager* models the moving space, including the dimension, shape and location of the field and the objects inside the field. It also keeps track of robots' *(x,y)* positions and moving directions during simulation. Such tracking is needed to supply robots with the correct sensory data. To account for variability in the real motion, a random number generator provides a source of additive noise. Note that in this example we have ignored the dynamics of a movement as we treat each movement as an atomic action so the positions and directions of robots are updated discretely.

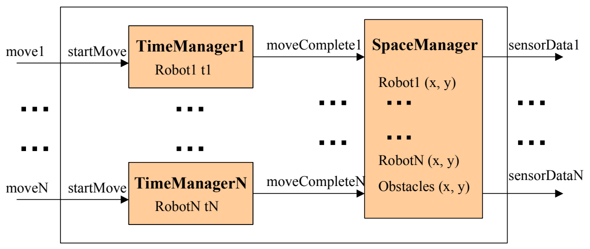
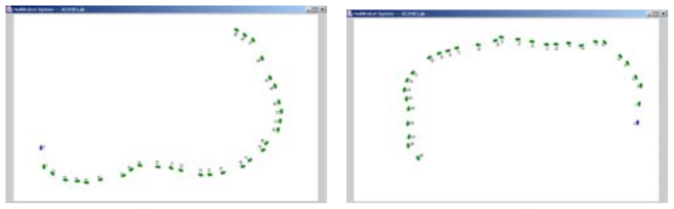


Figure 5: Environment model

With all these models, simulation was run and a graphic user interface was developed to show robots' movements. Figure 6 shows two snapshots of a robotic convoy system with 30 robots within a field surrounded by walls. As can be seen in this system, robots will not follow the exact track of the leader robot. However, they are able to follow their immediate front robots closely, thus forming a coherent team from a global point of view. Note that obstacles can also be easily added within the field.



(1) (2)

Figure 6: Snapshot of robots in motion

3.2 Measuring Metrics and Simulation Results

A good set of measuring metrics is very important to study this robotic convoy system. This section describes several metrics that we have developed. These metrics are neither final nor complete. However, they serve as a starting point to analyze and measure this system. Some preliminary simulation results based on these metrics are also presented and analyzed.

Convoy Speed and Number of Adjustment

The convoy speed of the team and the number of adjustment for each robot are among the most obvious results that can be obtained from simulation-based study. Both them can be viewed as metrics for the system's performance. In fact, these two metrics are correlated to each other: the larger the number of adjustment, the slower the convoy speed. Since robots move in a coordinated way, we define the convoy speed as the speed of the leader robot. This can be calculated by dividing the moving distance by the logic time of simulation. The number of adjustment can be obtained directly from each robot.

Formation Coherence

Due to noise and variance in reality, there exists difference between a robot's real position, direction (angle) and its desired position and direction. This difference is affected by the variance of movement in real execution, which is modeled by adding noise into robot's movement in simulation. On the other hand, even though variance exists, this system can still conduct the convoy with some level of formation coherence. This is because an "adjust process" has been implemented that allows robots to adjust their positions/directions based on the feedback from its infrared sensors. Apparently the level of formation coherence is affected by the variance of movement. If this variance is large enough, even though a adjust process exists, the system will eventually fail to maintain its formation coherence.

To study this problem, we calculate each robot's position errors under the effect of distance noise factor (DNF) and angle noise factor (ANF). These two factors are the ratio of the maximum distance variance and maximum angle variance as compared to the robot's moving distance respectively. For example, if the angle noise factor is 0.1 and a robot moves forward 60, after its movement the robot will have maximum 6 degrees variance from its desired direction. Once each robot's position error is known, the average position error of the team can be derived. This average is an indicator for the convoy system's formation coherence: the smaller the error is, the more coherent the convoy system is. Formula (4) – (7) shows how the average position error can be calculated. In these formulas, *D* is the desired distance between robots and *N* is the total number of robot. In case the formation coherence is broken, saying robot R_i lose itself, $E_i(t)$ will increase continuously, making the average error $E(t)$ increase too. Note that the desired position ($x_{i\text{-desired}}$, $y_{i\text{-desired}}$) of R_i is calculated from its front robot R_{i-1}'s position, not related to any specific formations. Thus systems with different line formation shapes may have the same position errors.

$$E_i(t) = \sqrt{(x_i(t) - x_{i-desired}(t))^2 + (y_i(t) - y_{i-desired}(t))^2} \qquad (4)$$

$$x_{i-desired}(t) = x_{i-1}(t) - D * \cos\theta_{i-1}(t) \qquad (5)$$

$$y_{i-desired}(t) = y_{i-1}(t) - D * \sin\theta_{i-1}(t) \qquad (6)$$

$$E(t) = \sum E_i(t) \Big/ N \qquad (7)$$

Figure 7 shows the average poison error for a system with 30 robots, DNF=0.1, and ANF=0.08. The system starts with all robots at their desired positions. Thus as simulation proceeds, the position error increases from 0. It then reaches a "stable" stage where the position error oscillates around an average value (35.7 In this example). As we can see, in this system the position error does not accumulate over time. Thus we say that this system's formation coherence is maintained.

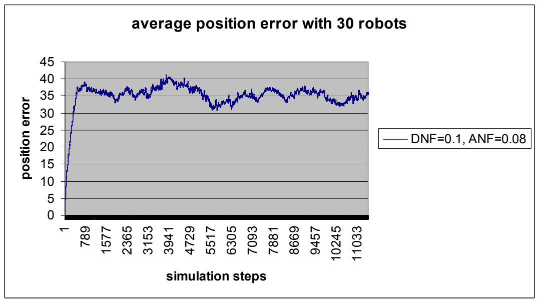


Figure 7: Average position error with 30 robots

Sensitivity

Since the formation coherence is affected by the noise factors, sensitivity analysis is useful to study if the system is robust to noise factors. To conduct sensitivity analysis, we run simulations with different noise factors and calculate the position errors. Figure 8 shows a system with 30 robots' average position errors under the effect of three sets of DNF and ANF: set 1 has DNF = 0.04, ANF = 0.04; set 2 has DNF =0.1, ANF = 0.08; set 3 has DNF = 0.2, ANF = 0.1. For analysis purpose, we omit the "transient " stage when the simulations start.

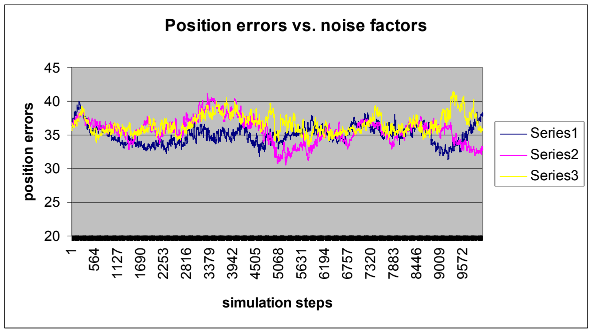


Figure 8: Average position errors vs. noise factors

Figure 8 shows that different noise factors result in different error patterns. However, for this system, all three errors are still maintained within a boundary (they do not accumulate as time increases). By calculating the average of them, we have average1 = 35.1, average2 =35.7, and average3 =36.6. From these data we can see that as the noise factor increases, the position error increases too. However, this change is insignificant as compared to change of the noise factors. Although more analysis is needed to reach any quantitative conclusion, we can say that this system is insensitive to the noise factors as long as these factors are within a safe boundary. This is because the system impalements an adjust process that allows robots to adjust themselves based on the feedback from their IR sensors.

Scalability

Scalability refers to the ability of a system to maintain its quality as the scale of the system increases. To study scalability, we change the number of robots and run simulation to see how that affects system's average position error (average over number of robots and over time). Figure 9 shows the position errors for the number of robots to be 10, 20, 30, and 40 with DNF =0.1 and ANF =0.08. It shows that the average position error increases as the number of robot increases. If this trend holds true with more robots, the system is not scalable in the sense that it will eventually break as more robots are added into the system.

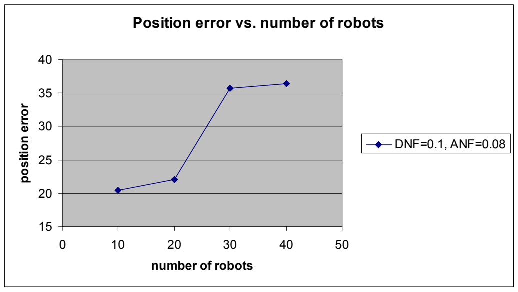


Figure 9: Average position errors vs. number of robots

4. CONCLUSION

This paper presents a simulation-based virtual evaluation environment for cooperative robotic systems. This virtual environment allows a combination of real and virtual robots to work together for a system-wide study and measurement. An incremental measuring process is developed to transition simulation-based study closer to reality as the process proceeds. Based on this virtual environment, a robotic convoy system was developed and presented in this paper as an illustrative example. Coherence metrics for this system were defined and preliminary simulation results were discussed.

We note that most results presented in this paper are collected from simulations that do not involve real robots. But in the next step we plan to measure the system using robot-in-the-loop simulation and expect to gather more interesting results. In the meantime, a set of more complete evaluation metrics is also under development for the robotic convoy systems presented in this paper.

5. ACKNOWLEDGEMENT

This research was supported by NSF grant DMI- 0122227, "DEVS as a Formal Framework for Scalable Enterprise Systems."

6. REFERENCE

[1] Dudek, G., Jenkin, M., and Milios, E., "A Taxonomy of Multirobot Systems", *Robot Teams*, Edited by Balch, T., and Parker L.E., A K Peters, 2002

[2] Balch, T., "Taxonomies of Multirobot Task and Reward", *Robot Teams*, Edited by Balch, T., and Parker L.E., A K Peters, 2002

[3] Hu, X., and Zeigler, B. P., "Model Continuity to Support Software Development for Distributed Robotic Systems: a Team Formation Example", *Journal of Intelligent & Robotic Systems, Theory & Application, Special Issue: Multiple and Distributed Cooperating Robots*, pp. 71-87, January, 2004

[4] Hu, X., and Zeigler, B.P., "A Simulation-based Software Development Methodology for Cooperative Real-time Intelligent Systems", to appear in *Annual of Complex Systems and Intelligence Science*, World Scientific Publishing Co., 2004

[5] Zeigler, B.P., Kim, T.G., *et al.*. Theory of Modeling and Simulation. New York, NY, Academic Press, 2000.

[6] Komoriya, K.; Tani, K., "Utilization of the virtual environment system for autonomous control of mobile robots", *Intelligent Motion Control, 1990. Proceedings of the IEEE International Workshop on*, Volume: 2, 20-22 August 1990

[7] Wang. J.: Methodology and design principles for a generic simulation platform for distributed robotic system experimentation and development. *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, Volume: 2, 1997 Page(s): 1245 -1250 vol.2

[8] Dixon, K.; Dolan, J.; Wesley Huang; Paredis, C.; Khosla, P., "RAVE: a real and virtual environment for multiple mobile robot systems", *Intelligent Robots and Systems, 1999. IROS '99. Proceedings*. 1999 IEEE/RSJ International Conference on , Volume: 3 , 17-21 Oct. 1999

[9] Shaw, S. C., Real-time Systems and Software, John Wiley & Sons, 2001

[10] Hu, X., *A Simulation-based Software Development Methodology for Distributed Real-time Systems*, Dissertation, University of Arizona, 2003

[11] Peipelman. J., N. Alvarez, K. Galinet, R. Olmos.: 498 A & B Technical Report. Department of Electrical and Computer Engineering, University of Arizona, 2002

[12] Brooks, R. A., "A Robust Layered Control System For A Mobile Robot", IEEE Journal Of Robotics And Automation, RA-2, April. pp. 14-23, March 1986